PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-128325
(43)Date of publication of application : 16.05.1997

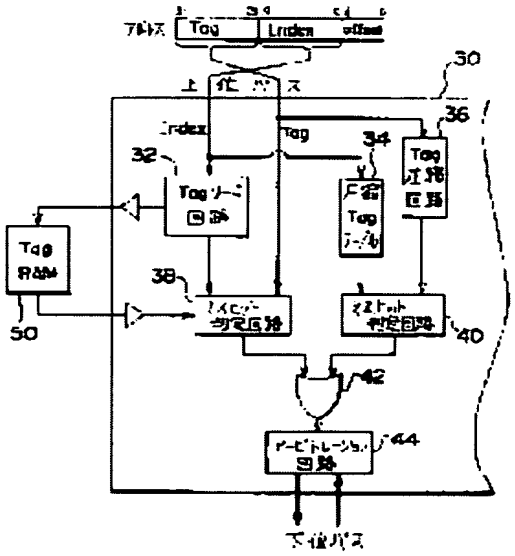
(51)Int.Cl. G06F 13/36
G06F 15/163
G06F 15/16

(21)Application number : 07-287372
(22)Date of filing : 06.11.1995

(71)Applicant : MITSUBISHI ELECTRIC CORP
(72)Inventor : KAMEMARU TOSHIHISA

(54) HIERARCHICAL BUS CONTROL SYSTEM AND BUS BRIDGE

(57)Abstract:
PROBLEM TO BE SOLVED: To reduce the latency at the time of transferring a transaction from one bus to the other bus by snooping a cache when the transaction is issued to one of a high-order bus or a low-order bus and precedingly executing the arbitration of the other bus before deciding whether or not the transaction is transferred to the other bus.
SOLUTION: When the transaction of a high-order bus is inputted to a bus bridge 30, the index part of an address in the transaction is inputted to a tag- read circuit 32 and a tag corresponding to the index is read from a tag RAM 50. The tag is compared with the tag of the actual address in a mis-hit judging circuit 38. At the time of non-coindidence, the mis-hit judging circuit 38 issues a mis-hit signal, it is inputted to an arbitration circuit 44 with an OR circuit 112 and the arbitration of a low-order bus is executed. After that, the transaction to be transferred is issued to the low-order bus.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of

rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平9－128325

(43)公開日　平成9年（1997）5月16日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ			技術表示箇所
Ｇ０６Ｆ	13/36	５３０		Ｇ０６Ｆ	13/36	５３０	Ｂ
	15/163				15/16	３２０	Ｋ
	15/16					４００	Ｂ

審査請求　未請求　請求項の数10　ＯＬ（全 14 頁）

(21)出願番号　特願平7－287372

(22)出願日　平成7年（1995）11月6日

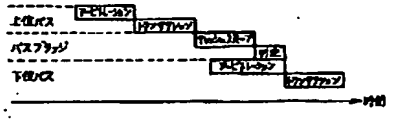
(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目2番3号

(72)発明者　亀丸　敏久
東京都千代田区丸の内二丁目2番3号　三菱電機株式会社内

(74)代理人　弁理士　吉田　研二　（外２名）

(54)【発明の名称】階層バス制御方式及びバスブリッジ

(57)【要約】
【課題】　階層バスシステムのレイテンシを低減する。
【解決手段】　バスブリッジは、上位バスからのトランザクションを受け取った場合、ブリッジキャッシュのスヌープが完了する前から下位バスの先行アービトレーションを開始する。トランザクションの種類又はアドレスに応じて、この先行アービトレーションを行うか否かを選択することもできる。



1

【特許請求の範囲】

【請求項1】　複数のプロセッサを上位バスに接続し、複数の上位バスに対してキャッシュを有するバスブリッジをそれぞれ設け、各上位バスを各々のバスブリッジを介して下位バスに接続し、この下位バスをメインメモリに接続した階層バスシステムにおいて、

前記バスブリッジは、上位バス又は下位バスの一方にトランザクションが発行されたときに、前記キャッシュをスヌープして他方のバスにトランザクションを転送するか否かを決定する前に、他方のバスのアービトレーションを先行して行うことを特徴とする階層バス制御方式。

【請求項2】　請求項1記載の階層バス制御方式において、

前記バスブリッジは前記一方のバスに発行されたトランザクションの種類又はアドレスに基づきそのトランザクションが前記他方のバスに転送される可能性を判定し、トランザクションが転送される可能性が高いと判定された場合にのみ、前記他方のバスに対する前記先行アービトレーションを行うことを特徴とする階層バス制御方式。

【請求項3】　請求項2記載の階層バス制御方式において、

前記バスブリッジは、上位バスから下位バスへトランザクションを転送したときに当該トランザクションのアドレスを記憶し、このトランザクションの後に、前記記憶されたアドレスが含まれるキャッシュブロックに連続したキャッシュブロック内のアドレスに対するトランザクションが上位バスに発行された場合に、後者のトランザクションは下位バスに転送される可能性が高いと判定することを特徴とする階層バス制御方式。

【請求項4】　請求項2記載の階層バス制御方式において、

前記バスブリッジは、下位バスにインバリデート（Ｉｎｖａｌｉｄａｔｅ）系トランザクションが発行された場合には、そのトランザクションは上位バスに転送される可能性が高いと判定することを特徴とする階層バス制御方式。

【請求項5】　スプリット対応可能な複数のプロセッサを上位バスに接続し、複数の前記上位バスに対してキャッシュを有するバスブリッジをそれぞれ設け、前記各上位バスを各々のバスブリッジを介して下位バスに接続し、この下位バスをメインメモリに接続した階層バスシステムにおいて、

前記バスブリッジは、上位バスのトランザクションをスプリットして下位バスへ転送した場合には、当該転送したトランザクションに対する下位バスからの応答を待たずに、前記転送処理から所定の時間間隔の後に上位バスに対するアービトレーションを先行して行うことを特徴とする階層バス制御方式。

【請求項6】　請求項5記載の階層バス制御方式において、

2

前記バスブリッジは、上位バスから下位バスへ転送したトランザクションの種類を検出し、その検出結果がインバリデート（Ｉｎｖａｌｉｄａｔｅ）系であった場合には、検出結果が非インバリデート系であった場合よりも上位バスのアービトレーションを行うまでの前記時間間隔を大きくすることを特徴とする階層バス制御方式。

【請求項7】　請求項1～6のいずれかに記載の階層バス制御方式において、

前記バスブリッジに接続されたプロセッサのバス占有率の総和が所定のしきい値を越える場合は前記上位バスに対する前記先行アービトレーションを禁止することを特徴とする階層バス制御方式。

【請求項8】　請求項1～6のいずれかに記載の階層バス制御方式において、

階層バスシステムに含まれるプロセッサ及びＩ／Ｏ装置のバス占有率の総和が所定のしきい値を越える場合は前記下位バスに対する前記先行アービトレーションを禁止することを特徴とする階層バス制御方式。

【請求項9】　請求項1～6のいずれかに記載の階層バス制御方式において、

前記バスブリッジは、当該バスブリッジに接続された上位バス及び下位バスのバス負荷をそれぞれ監視し、各バスのバス負荷が所定のしきい値を越えた場合に、しきい値を越えたバスについては前記先行アービトレーションを禁止することを特徴とする階層バス制御方式。

【請求項10】　複数のプロセッサが接続された上位バスとメインメモリが接続された下位バスとを接続するバスブリッジであって、タグ情報を格納したタグメモリとデータを格納したキャッシュメモリとから構成されるキャッシュユニットに接続され、一方のバスに発行されたトランザクションのアドレスに含まれるインデックスから前記タグメモリを検索し、検索の結果得られたタグが前記トランザクションアドレスに含まれるタグと一致しなかった場合に前記トランザクションアドレスがキャッシュメモリに含まれないと判定し、他方のバスに前記トランザクションを転送するバスブリッジにおいて、

前記タグメモリの情報を所定の方式で圧縮したテーブルを格納した圧縮タグメモリと、

前記一方のバスに発行されたトランザクションのアドレスからタグを抽出し前記所定の方式で圧縮するタグ圧縮手段と、

前記一方のバスに発行されたトランザクションのアドレスからインデックスを抽出し、そのインデックスに対応する圧縮されたタグ情報を前記圧縮タグメモリから読み出す圧縮タグ検出手段と、

前記タグ圧縮手段の出力と前記圧縮タグ検出手段の出力とを比較する比較手段と、

を有し、前記比較手段による比較の結果、前記タグ圧縮手段の出力と前記圧縮タグ検出手段の出力とが不一致の

ときには、前記タグメモリの検索の終了を待たずに前記他方のバスに対してアービトレーション信号を出力することを特徴とするバスブリッジ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、階層バスシステムにおいて、一方のバス（上位バス又は下位バス）から他方のバス（下位バス又は上位バス）へトランザクションを転送する際のバスの制御方式に関する。

【０００２】

【従来の技術】複数のＣＰＵをバスを介してメインメモリやＩ／Ｏ装置に共通接続し、他のＣＰＵやメインメモリ等とデータの授受を行うバス共有型のマルチプロセッサシステムが普及している。このマルチプロセッサシステムにおいて、演算能力の向上を狙って並列ＣＰＵ数を増やすと、バスへの負荷の増大するため結局スループットやレイテンシが悪化し、ＣＰＵ数の増加に見合うほどトータルの処理能力は向上しない。

【０００３】これに対し、本出願人は、特願平７－２８５８２９号にて、並列するＣＰＵを小数ずつそれぞれ別々の上位バスに接続し、各上位バスに対してそれぞれキャッシュメモリを有するバスブリッジを設け、各上位バスを各バスブリッジを介して、メインメモリ等が接続される下位バスに接続する階層構成のバスシステムを提案した。図５は、このような階層バスシステムの概略構成を示す図であり、ＣＰＵ１０は比較的小数ずつ１つの上位バス１２に接続されており、各上位バスはバスブリッジ１４を介してメインメモリ２０に接続された下位バス１８に接続されている。各バスブリッジ１４にはキャッシュ１６が接続されており、このキャッシュ１６は上位のＣＰＵ１０にて共有される。

【０００４】このような階層バスシステムによれば、ＣＰＵの総数が同じ場合、階層化しない場合に比べて上位バスに接続されるＣＰＵの数が少なくなるので、上位バスの負荷が小さくなる。また、バスブリッジにキャッシュを設けたことにより、上位バスのトランザクションの要求データがバスブリッジのキャッシュにある場合にはメインメモリまでデータを読みにいく必要がないので下位バスの負荷も大きいものにはならない。したがって、この階層構成のマルチプロセッサシステムによれば、多数のＣＰＵを効率的に動作させることが可能になる。

【０００５】

【発明が解決しようとする課題】このような階層バスシステムにおいて、上位バスあるいは下位バスにトランザクションが発行された場合、バスブリッジはキャッシュを検索（スヌープ）し、トランザクションを下位バスあるいは上位バスに転送する必要がないか判定する。例えば、一方のバスからのトランザクションの要求データがキャッシュにある場合にはバスブリッジはトランザクションの転送を行わず、キャッシュにない場合には他方のバスに対してトランザクションを転送する。

【０００６】このようにして一方のバスから他方のバスへトランザクションを転送することが決定された場合、バスブリッジは他方のバスのバス使用権を得るためにアービトレーションを行う必要がある。従来、バスブリッジからバスへのアービトレーションは、キャッシュのスヌープが完了しトランザクションを転送することが決定されてから行っていた。

【０００７】図６は、上位バスに発行されたトランザクションが下位バスに転送されるまでの従来の階層バスシステムの処理の流れを示す説明図である。図６に示すように、ＣＰＵは、上位バスにトランザクションを発行しようとする場合、まず上位バスのアービトレーションを行い、これによってバス使用権を獲得する。その後、ＣＰＵは上位バスにトランザクションを発行する。この上位バスに接続されているバスブリッジは、このトランザクションを検出し、このトランザクションのアドレス（トランザクションが要求するデータのアドレス）を有していないか、自らのキャッシュをスヌープする。そして、バスブリッジは、このスヌープの結果、すなわちバスブリッジ・キャッシュがトランザクションが要求するデータを有しているか否か等の情報に基づき、バスブリッジは上位バスのトランザクションを下位バスに転送するか否か判定する。そして、この判定においてトランザクションを下位バスに転送すべきであると判定された場合に、バスブリッジは、下位バスに対してバス要求信号を発して下位バスのアービトレーションを行い、バス使用権を獲得した後に下位バスに対して転送すべきトランザクションを発行していた。

【０００８】このように従来の階層バスシステムでは、バスブリッジキャッシュのスヌープにより一方のバスから他方のバスへトランザクションを転送することが決定されて初めて転送先バスに対するアービトレーションを行っていた。一方、バスブリッジのキャッシュは、上位バスに接続される複数のＣＰＵに対応するため比較的大容量すなわち比較的低速のものを用いる必要があるので、このキャッシュをスヌープするのにはある程度時間が掛る。このため、従来の階層バスシステムには、バスブリッジを介してトランザクションの転送を行う度に、処理の遅れ（レイテンシ）が生じるという問題があった。

【０００９】本発明は、このような問題を解決するためになされたものであり、階層バスシステムにおいて、上位バスから下位バスへ又は下位バスから上位バスへトランザクションを転送する際のレイテンシを少なくする階層バス制御方式を提供することを目的とする。

【００１０】

【課題を解決するための手段】前述の目的を達成するために、本発明に係る階層バス制御方式は、複数のプロセッサを上位バスに接続し、複数の上位バスに対してキャ

ッシュを有するバスブリッジをそれぞれ設け、各上位バスを各々のバスブリッジを介して下位バスに接続し、この下位バスをメインメモリに接続した階層バスシステムにおいて、前記バスブリッジは、上位バス又は下位バスの一方にトランザクションが発行されたときに、前記キャッシュをスヌープして他方のバスにトランザクションを転送するか否かを決定する前に、他方のバスのアービトレーションを先行して行うことを特徴とする。

【００１１】この構成によれば、一方のバスにトランザクションが発行されると、バスブリッジは、比較的時間の掛るバスブリッジキャッシュのスヌープが完了する前から他方のバスのアービトレーションを行うので、他方バスのアービトレーションの完了するタイミングが従来に比べて早くなる。したがって、キャッシュスヌープが終わって他方バスへのトランザクション転送が決定された場合には、バスブリッジは従来より早く他方バスへトランザクションを発行することができるので、一方のバスから他方のバスへトランザクションを転送する際のレイテンシを少なくすることができる。

【００１２】なお、この構成及び以下に示す構成おいて、バスブリッジが一方のバスから他方のバスへトランザクションを「転送する」とは、バスブリッジが、一方のバスから受け取ったトランザクションに対して、当該トランザクションの種類や前記キャッシュの状態等に基づいて決定したトランザクションを他方のバスに発行することを意味する。したがって、バスブリッジが他方のバスに発行するトランザクションが、バスブリッジが一方のバスから受け取ったトランザクションと同じ種類のトランザクションになるとは必ずしも限らない。

【００１３】また、本発明の別の構成は、前記構成において、前記バスブリッジが、前記一方のバスに発行されたトランザクションの種類又はアドレスに基づきそのトランザクションが前記他方のバスに転送される可能性を判定し、トランザクションが転送される可能性が高いと判定された場合にのみ、前記他方のバスに対する先行アービトレーション（すなわち、キャッシュをスヌープして他方のバスにトランザクションを転送するか否かを決定する前に、先行して行うアービトレーション）を行うことを特徴とする。

【００１４】この構成は、無駄な先行アービトレーションを低減することを目的とするものである。すなわち、もしスヌープの結果、一方のバスのトランザクションが要求するデータがバスブリッジキャッシュ内に存在した場合、結局トランザクションは他方のバスに転送されずスヌープの完了前に先行して行ったアービトレーションは無駄になってしまうが、この構成は、このような無駄を低減しようとするものである。ここで、一方のバスから他方のバスにトランザクションの転送が行われる可能性の高さは、トランザクションの種類又はアドレスからある程度推測できるので、この構成では、一方のバスに

発行されたトランザクションの種類又はアドレスからそのトランザクションの転送可能性を判定し、可能性が高いと判定された場合にのみ、先行アービトレーションを行う。したがって、この構成によれば、無駄な先行アービトレーションを低減し、バスの負荷を低減することができる。

【００１５】また、本発明の別の構成は、スプリット対応可能な複数のプロセッサを上位バスに接続し、複数の前記上位バスに対してキャッシュを有するバスブリッジをそれぞれ設け、前記各上位バスを各々のバスブリッジを介して下位バスに接続し、この下位バスをメインメモリに接続した階層バスシステムにおいて、前記バスブリッジは、上位バスのトランザクションをスプリットして下位バスへ転送した場合には、当該転送したトランザクションに対する下位バスからの応答を待たずに、前記転送処理から所定の時間間隔の後に上位バスに対するアービトレーションを先行して行うことを特徴とする。

【００１６】この構成は、上位バスがスプリットバスである場合において、上位バス上のプロセッサから発せられたトランザクションをバスブリッジがスプリットして下位バスへ転送した場合のレイテンシの短縮を目的とするものである。すなわち、本構成では、このスプリットしたトランザクションに対するスプリット応答を元のプロセッサに返送する際に必要な上位バスのアービトレーションを、そのトランザクションに対する下位バスから応答が来る前から予め行っておくことにより、上位バスにスプリット応答を発するタイミングを早める。この構成によれば、プロセッサから発せられたトランザクションがスプリットされた場合において、当該プロセッサにそのトランザクションに対するスプリット応答が戻るまでの時間を短縮することができる。

【００１７】なお、この構成において、トランザクションのスプリット転送のあと上位バスのアービトレーションを行うまでは、上位バスは当該元のプロセッサの占有から離れるので、その間、別のトランザクションのために上位バスを使用することができる。したがって、トランザクションのスプリット転送のあと上位バスのアービトレーションを行うまでの時間間隔は、プロセッサにどの程度まで上位バスの占有を許すかという点とスプリット応答のためのアービトレーションの遅れをどの程度まで許容するかという点とを総合的に考慮して決定する必要がある。

【００１８】また、本発明の別の構成は、バスブリッジに接続されたプロセッサのバス占有率の総和が所定のしきい値を越える場合は前記上位バスに対する前記先行アービトレーションを禁止することを特徴とする。

【００１９】この構成において、バス占有率とは、単位時間当たり１個の装置がバスを占有している時間のことである。プロセッサの場合には、バス占有率はそのプロセッサの性能から求めることができる。この構成によれ

ば、上位パスのパス負荷が高いと予測される場合には、上位パスの先行アービトレーションを行わないようにすることができる。

【0020】また、本発明の別の構成は、階層バスシステムに含まれるプロセッサ及びI／O装置のパス占有率の総和が所定のしきい値を越える場合は前記下位パスに対する前記先行アービトレーションを禁止することを特徴とする。

【0021】この構成によれば、下位パスのパス負荷が高いと予測される場合には、下位パスの先行アービトレーションを行わないようにすることができる。

【0022】また、本発明の別の構成は、各バスブリッジにて上位パス及び下位パスのパス負荷をそれぞれ常時監視し、各パスのパス負荷が所定のしきい値を越えた場合に、しきい値を越えたパスについては前記先行アービトレーションを禁止する事を特徴とする。

【0023】すなわち、先行アービトレーションが無駄になった場合、その先行アービトレーションの結果占有されていたパスが無駄になってしまう。例えば、パスが混んでいる場合などには、このような無駄なパス占有はスループットの低下を招く。そこで、この構成では各パスのパス負荷を常時検出し、パス負荷が所定のしきい値を越えた場合には当該パスに対する先行アービトレーションを禁止する。すなわち、この構成では、先行アービトレーションを許可するモードと禁止するモードとを動的に切り換えることにより、パス負荷の増加を防ぎ、スループットの低下を防止する。

【0024】また、本発明に係るバスブリッジは、タグメモリの情報を所定の方式で圧縮したテーブルを格納した圧縮タグメモリと、一方のパスに発行されたトランザクションのアドレスからタグを抽出し前記所定の方式で圧縮するタグ圧縮手段と、一方のパスに発行されたトランザクションのアドレスからインデックスを抽出し、そのインデックスに対応する圧縮されたタグ情報を前記圧縮タグメモリから読み出す圧縮タグ検出手段と、タグ圧縮手段の出力と前記圧縮タグ検出手段の出力とを比較する比較手段とを有し、前記比較手段による比較の結果、前記タグ圧縮手段の出力と前記圧縮タグ検出手段の出力とが不一致のときには、前記タグメモリの検索の終了を待たずに前記他方のパスに対してアービトレーション信号を出力することを特徴とする。

【0025】この構成において、トランザクションのインデックスに基づき圧縮タグメモリから読み出された圧縮タグと、そのトランザクションのタグ自身を圧縮したものとが不一致の場合には、当該キャッシュ内にはそのトランザクションの要求するデータは存在しない。この圧縮タグ同士の比較は、タグメモリの検索（スヌープ）よりも短時間で行うことができる。したがって、この構成によれば、比較的時間を要するタグメモリ検索の結果が得られる前に、圧縮タグの比較に基づきトランザクションの要求データが存在しないことを知ることができるので、トランザクションが転送されることをより早く知ることができる。よって、この構成では、タグ圧縮手段の出力と圧縮タグ検出手段の出力とが不一致のときに、バスブリッジが転送先パスにアービトレーション信号を出力することにより、アービトレーションの完了が早くなり、トランザクションの転送に要する時間を短縮することができる。

【0026】

【発明の実施の形態】以下、本発明の好適な実施形態を図面に基づいて説明する。

【0027】以下の各実施形態は、例えば図5に示した階層バスシステムに適用される。なお、以下の各実施形態においては、CPUは、自分自身専用のキャッシュ（以下、CPUキャッシュと呼ぶ）を有しているものとして説明する。また、各実施形態において、CPUキャッシュ及びバスブリッジのキャッシュ（以下、ブリッジキャッシュと呼ぶ）はライトバック方式のキャッシュであり、各キャッシュの内容の一貫性（キャッシュ・コンシステンシ）を維持するために、MESIプロトコルを用いてCPUキャッシュとブリッジキャッシュとの間のマルチレベル包含性（MLI）を維持している。

【0028】MLIとは、ある上位パスに接続されたCPUキャッシュにあるデータは、その上位パスに接続されたバスブリッジのブリッジキャッシュにも存在するということをいう。したがって、以下の実施形態では、ブリッジキャッシュは、少なくともその上位のCPUキャッシュの容量の総和よりも大きな容量を有している。

【0029】また、MESIプロトコルとは、各キャッシュブロックに対してそのブロックの状態（ステート）を示すデータを設けることによりキャッシュコンシステンシを保つ方式の一つであり、M，E，S，Iの4種類のステートを用いて各キャッシュブロックを制御する。ここで、M，E，S，Iの各ステートの示す意味は以下の通りである。

【0030】M(Modify)：そのブロックの更新されたデータ（すなわち、そのブロックの最新のデータ）をそのキャッシュだけが持つ。

【0031】E(Exclusive)：そのブロックに関しメインメモリと同一内容のデータをそのキャッシュだけが持つ。

【0032】S(Shared)：そのブロックに関しメインメモリと同一内容のデータを持つが、他のキャッシュも当該データを持っている可能性がある。

【0033】I(Invalid)：そのブロックは無効である。

【0034】なお、以上のステートは基本的には同レベルにあるキャッシュ同士の関係を示すものである。すなわち、CPUキャッシュのステートは、他のCPUキャッシュとの関係を示すものであり、ブリッジキャッシュ

のステートは、他のブリッジキャッシュとの関係を示すものである。したがって、ブリッジキャッシュのステートがMだったとしても、それは当該ブリッジキャッシュ自体が最新のデータを有することを意味するとは限らず、当該ブリッジキャッシュ自体かその上位のＣＰＵキャッシュのいずれかが最新のデータを持つことを意味することになる。

【００３５】以下、このようなプロトコルを採用するシステムを例にとって、本発明の実施形態について説明する。ただし、これはあくまで一例であって、本発明は上記以外のプロトコルを採用するシステムにも有効である。

【００３６】実施形態１．本実施形態は、上位バス又は下位バスのいずれか一方にトランザクションが発行された場合において、ブリッジキャッシュをスヌープして他方のバスにトランザクションを転送するか否かを決定する前に、先行して他方のバスのアービトレーションを行うことにより、トランザクション転送によって生じる処理の遅れを短縮するものである。

【００３７】以下、上位バスから下位バスへのトランザクション転送を例にとり、図１を参照して本実施形態の方法を説明する。

【００３８】図1に示すように、ＣＰＵは、上位バスにトランザクションを発行しようとする場合、まず上位バスのアービトレーションを行い、これによってバス使用権を獲得する。その後、ＣＰＵは上位バスにトランザクションを発行する。この上位バスに接続されているバスブリッジは、このトランザクションを検出し、このトランザクションのアドレス（トランザクションが要求するデータのアドレス）を有していないか、自らのキャッシュをスヌープする。

【００３９】従来は、このスヌープの結果に基づき上位バスから下位バスへトランザクションを転送する必要があると判定された後に下位バスのアービトレーションを開始していたが、本実施形態では、ブリッジキャッシュのスヌープが完了する前から下位バスのアービトレーションを開始する。すなわち、本実施形態では、バスブリッジは、上位バスからのトランザクションを受信すると、ブリッジキャッシュのスヌープを開始すると共に下位バスに対してアービトレーション信号（バス要求信号）を出力して、下位バスのアービトレーションを開始する。ここで、下位バスのアービトレーションの方式は、集中アービトレーション、分散アービトレーションのいずれでもよい。

【００４０】このトランザクション転送の決定の前にアービトレーションを先行して行う方式（以下、先行アービトレーションと呼ぶ）によれば、例えばブリッジキャッシュのスヌープが終了してトランザクションを下位バスへ転送することが決定される前に下位バスのアービトレーションが終了していれば、トランザクション転送が決定されると同時にバスブリッジから下位バスに対して転送すべきトランザクションを発行することができる。また、もしトランザクションを下位バスへ転送すると決定される前に下位バスのアービトレーションが完了しなくても、従来よりもアービトレーションを開始のタイミングが早いため、いずれにしても下位バスへのトランザクションの発行タイミングが従来よりも早くなる。したがって、本実施形態によれば、上位バスから下位バスへトランザクションが転送される場合の処理時間を短縮することができるので、バスシステム全体としてレイテンシを小さくすることができる。

【００４１】なお、本実施形態において、上位バスからのトランザクションの受信した後キャッシュスヌープ完了（すなわち、トランザクションの転送・非転送の決定）までの間に、いつ下位バスのアービトレーションを開始するかは、バスの負荷状態とトランザクション転送処理の時間短縮要求とのトレードオフを勘案して決定する。例えば、アービトレーションの開始のタイミングを早くすると、下位バスへのトランザクションの発行タイミングが早くなるかわりに、スヌープの完了の前にアービトレーションが終了した場合などにはバスブリッジが下位バスを無駄に占有する時間が生じてしまう。そこで、下位バスのバス負荷が大きい場合などには、バスブリッジが下位バスを無駄に占有する時間を短縮あるいは無くすために、アービトレーションの開始を遅くする。逆に、バス負荷が小さい場合には、アービトレーションの開始タイミングを早くすることにより、処理時間の短縮の効果を高めることができる。

【００４２】以上の例では、上位バスから下位バスへのトランザクション転送を例にとって説明したが、下位バスから上位バスへのトランザクション転送の場合も、上記の例と全く同様に処理時間を短縮することができる。

【００４３】次に実施形態１の変形例について説明する。以上説明した実施形態１では、一方のバスにトランザクションが発行された場合、バスブリッジは基本的に他方のバスの先行アービトレーションを行うので、ブリッジキャッシュのスヌープの結果トランザクションの要求データがブリッジキャッシュ内に存在した場合などには、トランザクションは他方のバスに転送されず、先行アービトレーションは無駄になってしまう。この変形例は、このような無駄な先行アービトレーションを少なくするための方式である。

【００４４】この方式では、バスブリッジが、一方のバスに発行されたトランザクションの種類又はアドレスに基づき、そのトランザクションが他方のバスに転送される可能性を判定し、トランザクションが転送される可能性が高いと判定された場合にのみ、他方のバスに対する先行アービトレーションを行う。

【００４５】バスブリッジにおけるトランザクション転送の可能性の判定の方法としては、例えば以下に示すも

のがある。

【0046】まず第1は、上位バスから下位バスへのトランザクション転送に際して、そのトランザクションのアドレスに基づき転送可能性を判定する方法である。この方法は、次の原理に基づくものである。

【0047】一連の処理を表す命令コード群はメインメモリ上の連続したアドレスに格納される可能性が高い。したがって、CPUが新しい命令コード群を読み出す場合、ブリッジキャッシュのミスヒットは連続して起こりやすい。すなわち、一連の命令コード群の読出しの場合、CPUからはその命令コード群が格納された一連のアドレスに対する一連のリードトランザクションが順次発行される（ただし、ここではCPUは自分専用のキャッシュを有しているので、実際にCPUから発行されるリードトランザクションのアドレスはキャッシュブロック単位で連続したものとなる）ので、あるトランザクションに対してブリッジキャッシュでミスヒットした場合、その次のトランザクションがミスヒットする可能性は高い。一方、連続したトランザクションがキャッシュブロック単位で連続したアドレスに対するトランザクションである場合には、それら一連のトランザクションは、一連の命令コード群に対するものである可能性が高い。したがって、この第1の方法では、上位バスから下位バスにトランザクション転送した際の要求アドレスをバスブリッジに記録しておき、そのバスブリッジに次に与えられたトランザクションのアドレスが前回記録されたアドレスとキャッシュブロック単位で連続しているか否かを判定し、連続している場合には後のトランザクションが下位バスに転送される可能性は高いと判定する。なお、この第1の方法において、バスブリッジにおける転送トランザクションのアドレス記録は、転送を行う度に更新する。

【0048】トランザクション転送の可能性の判定の第2の方法は、下位バスから上位バスへのトランザクション転送において、そのトランザクションの種類に基づき転送可能性を判定する方法である。バス上に発行されるトランザクションには、リード（Ｒｅａｄ），インバリデート（Ｉｎｖａｌｉｄａｔｅ），リード・アンド・インバリデート（Ｒｅａｄ＆Ｉｎｖａｌｉｄａｔｅ）などがある。ここで、リードはCPUが自分のキャッシュにないデータを読み出す際に発行されるトランザクションであり、インバリデートはCPUが自分のキャッシュに持っているデータを書き替える際に、他のキャッシュとのコンシステンシを維持するために、他のキャッシュ内の該当ブロックを無効化するためのトランザクションである。また、リード・アンド・インバリデートは、CPUが自分のキャッシュにないデータについてライトを行う際に発行されるトランザクションであり、該当データを含むブロックの読み出しと他のキャッシュにおける該当ブロックの無効化とを同時に指示するものである。このようなトランザクションのうち、インバリデートやリード・アンド・インバリデートなど他のキャッシュの無効化を指示するトランザクションは、インバリデート系トランザクションと総称される。第2の方法では、このようなトランザクションの種類をバスブリッジで判別し、トランザクション転送の可能性を判定する。

【0049】すなわち、この第2の方法は、下位バスに発行されたトランザクションをバスブリッジが受け取った場合、バスブリッジにてそのトランザクションの種類を特定し、そのトランザクションがインバリデート系であった場合には先行アービトレーションを行い、そのトランザクションが非インバリデート系の場合には先行アービトレーションを行わないというものである。このような方法を採るのは、次の理由からである。

【0050】あるCPUにおいてあるアドレスに対するデータライト（書き替え）動作を行う場合、キャッシュコンシステンシを保つためには、同一上位バス上のCPUだけでなく、バスブリッジや他の上位バス上のCPUにも該当ブロックの無効化を指示する必要がある。このような場合、そのCPUの接続されたバスブリッジから下位バスに対しても、インバリデート系のトランザクションが発行される。このようにして下位バスにインバリデート系トランザクションが発行された場合、それを受け取ったバスブリッジでは、ブリッジキャッシュのステートがM，E，Sのいずれかであるとそのバスブリッジの上位のCPUが該当データを持っている可能性があるため、バスブリッジから上位バスに対してインバリデート系のトランザクションを発行しなければならない。

【0051】これに対して、下位バスに発行されたトランザクションが非インバリデート系（例えば、リード）であった場合において、キャッシュコンシステンシの維持が必要となるのは、そのトランザクションを受け取ったバスブリッジのキャッシュのステートがMの場合だけである。非インバリデート系トランザクションでは、他のCPUが最新のデータ（すなわち、ステートM）のデータを持っている場合にのみ、その最新データを読み出す必要があるため、上位バスに対してそのトランザクションを転送する必要がある。

【0052】ここで、ブリッジキャッシュのあるブロックのステートがMである可能性は、M，E，Sのいずれかである可能性よりも当然に低い。しかも、ステートがMとなるのはデータが書き替えられた場合であるので、命令コードのブロックがMになることはあまり考えられない。以上を総合すれば、下位バスのトランザクションが上位バスへ転送される可能性は、インバリデート系の方が非インバリデート系よりもはるかに高いといえる。

【0053】以上のような理由から、この第2の方法では、下位バスに発行されたトランザクションがインバリデート系の場合は、上位バスに対してインバリデート系のトランザクションを発行、すなわち転送しなければな

らない可能性が高いと判定する。

【0054】以上、トランザクションの種類又はアドレスからトランザクション転送の可能性を判定する方法の例をいくつか挙げたが、トランザクション転送の可能性の判定方法は以上の例にあげたものに限られるものではない。いずれにしても、トランザクション転送の可能性の高低を、キャッシュスヌープに要する時間よりも短時間で判定することができる方法であれば、どのような方法を用いてもよい。

【0055】そして、このような判定方法によって一方のバスから他方のバスへトランザクションが転送される可能性を判定し、転送可能性が高いと判定された場合にのみ他方のバスに対して先行アービトレーションを行うことにより、無駄な先行アービトレーションの数を低減することができる。

【0056】以上、トランザクションの種類やアドレスに基づき先行アービトレーションを行うか否かを決定することにより、無駄な先行アービトレーションを低減する方法について説明した。ただし、トランザクション転送先のバスのバス負荷が小さい場合は、無駄な先行アービトレーションを行ってもスループットにはあまり影響がないので、実施形態１の実施に当たって上記変形例の方法は必須のものではない。

【0057】実施形態２．本発明の実施形態２は、上位バスがスプリットバスである場合において、スプリット応答時のアービトレーションによるレイテンシを低減することを目的とする。

【0058】階層バスシステムにおいて上位バスにスプリットバスを採用するのは、上位バスのスループットを向上させるためである。すなわち、ＣＰＵから上位バスにトランザクションが発せられた場合において、ブリッジキャッシュに該当データが存在せず下位バスにトランザクションを転送した場合、下位バスからそのトランザクションに対する応答が返ってくるまでの間、上位バスは元のＣＰＵに占有されたままになる。これでは、その間に他のＣＰＵはトランザクションを発することができず、上位バスのスループットは低くなる。これに対して、ＣＰＵをスプリット対応可能とし上位バスをスプリットバスとして構成すれば、上位バスのトランザクションを下位バスへ転送する場合にはそのトランザクションをスプリットして元のＣＰＵにバスを放棄させることができ、その間に他のトランザクションを発行することが可能となる。したがって、この場合、上位バスのスループットは向上する。近年、スプリットに対応可能なＣＰＵが開発されており、これを用いれば上位バスのスプリットを実現することができる。

【0059】上位バスのトランザクションをスプリットして下位バスに転送した場合、そのトランザクションに対する下位バスからの応答をバスブリッジを介して元のＣＰＵに返す必要がある。バスブリッジから元のＣＰＵに対するこのような応答のことをスプリット応答という。このスプリット応答を行う場合には、バスブリッジはアービトレーションを行って上位バスの使用権を獲得する必要がある。なぜなら、スプリットを行った時点で元のＣＰＵは上位バスの使用権を放棄しているからである。

【0060】このスプリット応答時のアービトレーションは、例えば図２に示すように、転送したトランザクションに対する下位バスからのレスポンス（応答）をバスブリッジが受けた後に行う方法も考えられるが、これではアービトレーションを待つ分だけスプリット応答の出力タイミングが遅くなってしまう。

【0061】そこで、本実施形態では、バスブリッジが上位バスのトランザクションをスプリットした場合、下位バスからバスブリッジに対する応答を待たずに、スプリット時点から所定時間間隔の後にそのバスブリッジが上位バスのアービトレーションを行うようにする。

【0062】このときの処理の流れを示す説明図が図３である。図３において、上位バスに発行されたトランザクションについて、バスブリッジのキャッシュスヌープの完了前に下位バスのアービトレーションを行うところまでは、図１の処理の流れと同様である。ただし、この実施形態ではトランザクションのスプリットを行うため、バスブリッジは、スヌープの結果トランザクションの転送を決定した場合に元のＣＰＵに対してスプリットしたことを示す旨のレスポンスを返す点が、図１の場合と異なる。このレスポンスを受け取った元のＣＰＵは、上位バスを放棄し、バスブリッジからのスプリット応答がくるのを待ち受ける。

【0063】下位バスでは、転送されたトランザクションについて他のバスブリッジに対するスヌープが行われ、このスヌープの結果に基づいて、当該トランザクションに対するレスポンスが生成される。そして、このレスポンスを受け取ったバスブリッジは、スプリット応答を生成し上位バスに対して出力する。

【0064】本実施形態では、このスプリット応答のために必要なバス使用権の獲得のため、バスブリッジに下位バスのレスポンスが返ってくるのを待たずに上位バスのアービトレーションを行う。すなわち、本実施形態では、スプリット応答のための上位バスアービトレーションを、下位バスからのレスポンスを契機として開始するのではなく、バスブリッジがキャッシュスヌープの結果トランザクションのスプリット転送を決定した時点から所定の時間間隔の後に開始する。本実施形態では、スプリット応答のための上位バスアービトレーションを、下位バスからのレスポンスを待たずに開始することを先行アービトレーションと呼ぶ。

【0065】この時間間隔は、上位バスアービトレーションの開始が少なくとも下位バスからのレスポンスよりも早くなるように設定される。この時間間隔を小さくす

るほど、上位パスアービトレーションの完了のタイミングが早くなるため、パスブリッジからのスプリット応答の発行のタイミングを早くすることができる。ただし、この時間間隔をあまり小さく設定するとパスブリッジが上位パスを無駄に占有する時間が生じてしまうので、下位パスに対するスプリット転送からレスポンスまでに要する時間と、上位パスのアービトレーションに要する時間とを考慮して最適な時間間隔を設定する。

【００６６】なお、スプリット転送から上位パスアービトレーションの開始までの所定時間間隔の間は、各ＣＰＵは別のトランザクションを上位パスに発行することができる。

【００６７】このように、本実施形態によれば、スプリット応答のための上位パスアービトレーションの完了タイミングを早くすることができるので、スプリット応答の発行タイミングを早くすることができ、ＣＰＵが最初にトランザクションを発行してからスプリット応答が完了するまでの一連のトランザクション処理全体に要する時間を短くすることができる。

【００６８】なお、この実施形態２において、スプリット転送の結果下位パスに発行されるトランザクションの種類によって前記所定時間間隔を変えるようにすれば、より効率的に先行アービトレーションを行うことができる。すなわち、下位パスのスヌープに要する時間は一般にトランザクションの種類により異なってくるので、下位パススヌープに要する時間によって上位パスアービトレーションの開始時刻を変えることにより、各トランザクション種類ごとに最適なタイミングで上位パスアービトレーションを実行することが可能になる。

【００６９】実施形態１の変形例にて説明したように、下位パスのトランザクションが上位パスへ転送される可能性は、インバリデート系の方が非インバリデート系よりも高い。したがって、スプリット転送の結果下位パスに発行されたトランザクションがインバリデート系である場合は、他のパスブリッジがその上位パスに対してトランザクションを発行する可能性が高い。よって、インバリデート系トランザクションの場合、下位パスのスヌープの結果は他のブリッジの上位パスにおける処理が終わらないと決定されないので、図３における下位パススヌープに要する時間は、非インバリデート系トランザクションに比べて長くなりがちである。

【００７０】したがって、スプリット転送から上位パスアービトレーションの開始までの所定時間間隔の値として、インバリデート系トランザクションに対するものと非インバリデート系トランザクションに対するものとで別々の値を設定し、インバリデート系に対する値を非インバリデート系に対する値よりも大きくすることにより、各トランザクション種類に応じて適切なタイミングでアービトレーションを行うことができる。なお、この場合、パスブリッジでは、下位パスにスプリット転送されるトランザクションの種類を常時検出する。

【００７１】以上、本発明に係る階層バス制御方式の第１及び第２の実施形態について説明したが、以上に説明した各方式をソフトウエアからの構成制御で切り換えるようにすれば柔軟な階層バスシステム構成が可能となる。すなわち、上記各実施形態の方式は具体的にはパスブリッジの機能として実現されるが、このパスブリッジを構成する際に前述の複数の先行アービトレーション方式のための手段を組み込んでおき、これら各方式および先行アービトレーションを禁止するモードを構成制御にて切り換えるのである。このような切り換えによれば、例えば、あるパスブリッジは上位パス・下位パスの両方について常に先行アービトレーションを行い、別のパスブリッジは下位パスについてトランザクションのアドレスに基づく先行アービトレーションを行い、上位パスについては先行アービトレーションを行わないなど、各パスブリッジごとにそれぞれ最適の先行アービトレーション方式を実行することができる。

【００７２】この切り換え方式において、先行アービトレーションを禁止するモードに切り換え可能とするのは、無駄な先行アービトレーションによるスループットの低下を防止するためのである。

【００７３】この禁止モードの選択は、例えばバス負荷の予測に基づいて行う。すなわち、上位パスについては、例えば、

T_{cpu} ×（ＣＰＵ個数）＞しきい値

となる場合に禁止モードを選択する。ここで、T_{cpu} はＣＰＵのバス占有率（単位時間当たりのＣＰＵのバス占有時間）であり、この値はＣＰＵの性能から予測できる。またＣＰＵ個数は、パスブリッジ自体の上位パスに接続されているＣＰＵの個数である。

【００７４】また、下位パスについては、例えば、

T_{cpu} ×（ＣＰＵ個数）＋$T_{i/o}$ ×（Ｉ／Ｏ個数）＞しきい値

となる場合に禁止モードを選択する。ここで、$T_{i/o}$ はＩ／Ｏ装置のバス占有率（単位時間当たりのＩ／Ｏ装置のバス占有時間）であり、この値はＩ／Ｏ装置の性能から予測できる。なお、この式におけるＣＰＵ個数、Ｉ／Ｏ個数は階層バスシステム全体についての個数である。

【００７５】このように、各パスブリッジごとに、バス負荷が大きくなると予測されるバスについては先行アービトレーションを禁止することにより、バスシステムの構成に応じてスループットとレイテンシを最適化することができる。

【００７６】また、パスブリッジにて上位パス及び下位パスのバス負荷を監視し、その監視結果に基づき動的に先行アービトレーションの許可・禁止を切り換えることもできる。すなわち、パスブリッジにバス負荷を監視する構成を設け、各パスのバス負荷が所定のしきい値より大きくなった場合に先行アービトレーションを禁止す

る。ここで、バス負荷は、例えば一定期間中に発生したバス要求の数から求める。この動的な先行アービトレーション制御によれば、階層バスシステムのスループットの低下を防止することができる。

【００７７】実施形態３．図４は、本発明の実施形態３に係るバスブリッジの要部構成を示すブロック図である。実際のバスブリッジには様々な回路が含まれるが、図４にはそのうちの先行アービトレーションに関する構成のみを示している。なお、図４は、上位バスからのトランザクションについての処理系のみを示している。

【００７８】この例では、アドレスは３２ビットで指定され、そのうちの上位１２ビットはタグ（Tag）、その次の１５ビットはインデックス（Index）として用いられ、下位の５ビットがオフセット（Offset）となっている。すなわち、この例では、４Gバイトのアドレス空間とブリッジキャッシュとは１キャッシュブロック＝３２バイト単位でマッピングされており、ブリッジキャッシュは10^{15}ブロックすなわち１Mバイトの容量を有している。なお、ブリッジキャッシュは、１Mバイトのデータを格納するキャッシュＲＡＭと、キャッシュＲＡＭの各ブロックに格納されているデータのアドレスをインデックスとタグとの対応関係の形で記憶するタグＲＡＭとから構成されているが、図４ではそのうちタグＲＡＭのみが図示されている。なお、ブリッジキャッシュはこのように大容量であるため、バスブリッジ本体とは別のチップとして形成されている。

【００７９】図４に示す構成は、通常のキャッシュ検索（スヌープ）のための構成と、先行アービトレーションのための構成とからなっている。

【００８０】通常のキャッシュスヌープは、タグリード回路３２、タグＲＡＭ５０及びミスヒット判定回路３８によって行われる。すなわち、上位バスのトランザクションがバスブリッジ３０に入力されると、トランザクションのアドレスのインデックス部分は、タグリード回路３２に入力される。タグリード回路３２は、タグＲＡＭ５０から、このインデックスに対応するタグを読み出す。タグＲＡＭ５０から出力されたタグは、ミスヒット判定回路３８にて実際のアドレスのタグと比較される。両者が一致した場合は、キャッシュＲＡＭ内にトランザクションの要求データが存在しており、この場合バスブリッジは、図示しないキャッシュコントローラを介してキャッシュＲＡＭから所望データを読み出し、上位バスを介して要求元のＣＰＵに返送する。この場合、下位バスにはトランザクションが転送されない。これに対して両者が不一致となった場合には、ミスヒット判定回路３８はミスヒット信号を発し、このミスヒット信号がオア回路４２を介してアービトレーション回路４４に入力される。この場合、アービトレーション回路４４は、下位バスに対してアービトレーション信号を発し、下位バスのアービトレーションを行う。その後、転送すべきトランザクションが下位バスに発行される。

【００８１】以上が、通常のキャッシュスヌープの流れであるが、本実施形態においてブリッジキャッシュは大容量でバスブリッジ３０とは別チップとして構成されているため、タグＲＡＭ５０の検索には比較的長時間を要する。

【００８２】一方、先行アービトレーションのための構成は、圧縮タグテーブル３４と、タグ圧縮回路３６及びミスヒット判定回路４０とからなる。圧縮タグテーブル３４には、タグＲＡＭ５０に格納されたインデックスとタグの対応情報を圧縮した情報が格納されている。すなわち、圧縮タグテーブル３４には、各インデックスごとに、対応するタグを所定のデータ圧縮法で圧縮した値（圧縮タグと呼ぶ）が格納される。データ圧縮法としては、例えばタグの全ビットの排他的論理和をとるなどの方法を用いる。圧縮タグテーブル３４は、タグＲＡＭ５０に比べてはるかに小さいサイズで済むため、バスブリッジ３０のチップ内に搭載することが可能であり、高速アクセスが可能である。圧縮タグテーブル３４は、トランザクションのアドレスのインデックスが与えられると、そのインデックスに対応する圧縮タグを出力する。一方、タグ圧縮回路３６は、トランザクションアドレスのタグを圧縮タグテーブル３４と同様のデータ圧縮法で圧縮する。

【００８３】ミスヒット判定回路４０は、圧縮タグテーブル３４の出力とタグ圧縮回路３６の出力とを比較する。ここで、圧縮タグ同士が不一致の場合はもともとのタグ同士も必ず不一致になるので、ミスヒット判定回路４０での比較結果が不一致となる場合は、当該トランザクションは必ず下位バスに転送される。そこで、ミスヒット判定回路４０は、両入力の値が不一致であったときにミスヒット信号を出力することにより、アービトレーション回路４４に対して下位バスのアービトレーションを指示する。なお、このミスヒット信号は、オア回路４２を介してアービトレーション回路４４に入力される。

【００８４】ここで、圧縮タグテーブル３４の検索はタグＲＡＭ５０の検索よりも短時間で済むため、ミスヒット判定回路４０のミスヒット信号はミスヒット判定回路３８のミスヒット信号よりも早く出力される。したがって、本構成によれば、圧縮タグ同士の比較によりトランザクション転送が確実に起こる場合を予め検知することができるので、キャッシュスヌープ完了前に下位バスの先行アービトレーションを開始することができる。

【００８５】なお、図４の構成では、ミスヒット判定回路４０からミスヒット信号が出力される場合には必ずトランザクション転送が起こるので、無駄な先行アービトレーションは生じない。

【図面の簡単な説明】

【図１】　実施形態１におけるトランザクション転送のための先行アービトレーションを説明するための図であ

る。

【図2】 スプリット応答のための先行アービトレーションを説明するための図である。

【図3】 実施形態2におけるスプリット応答のための先行アービトレーションを説明するための図である。

【図4】 実施形態3に係るバスブリッジの要部構成を示すブロック図である。

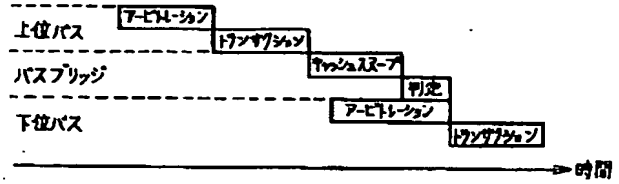
【図5】 階層バスシステムの構成を示す説明図である。

【図6】 従来の上位バスから下位バスへのトランザクションの転送処理を説明するための図である。

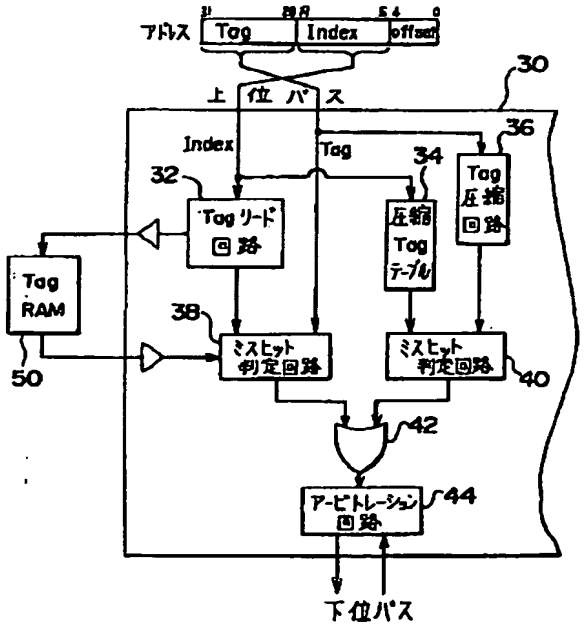
【符号の説明】

10 CPU、12 上位バス、14 バスブリッジ、16 キャッシュ、18 下位バス、20 メインメモリ、22 I／O装置。

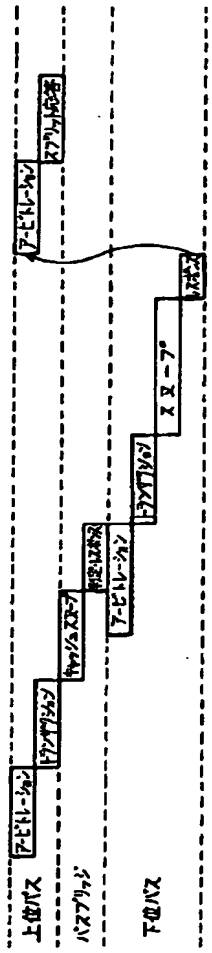
【図1】



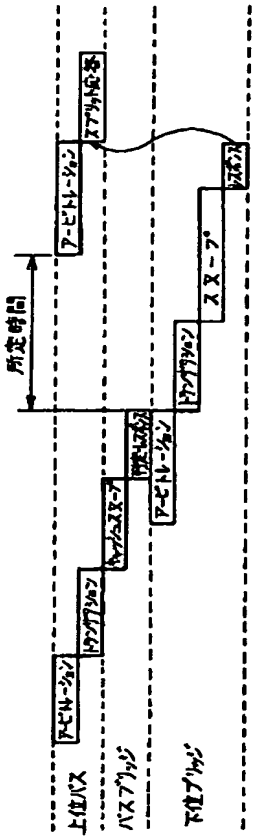
【図4】



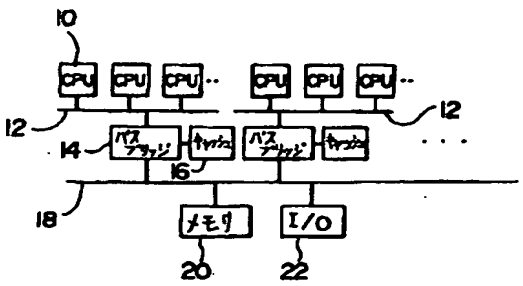
【図2】



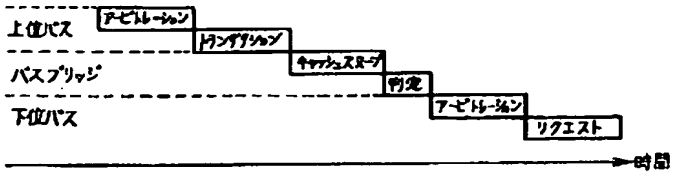
【図3】



【図5】


10
CPU
CPU
CPU
CPU
CPU
CPU
12
12
14
16
18
メモリ
I/O
20
22

【図6】


上位バス
アービトレーション
トランザクション
キャッシュスヌープ
判定
アービトレーション
リクエスト
バスブリッジ
下位バス
時間